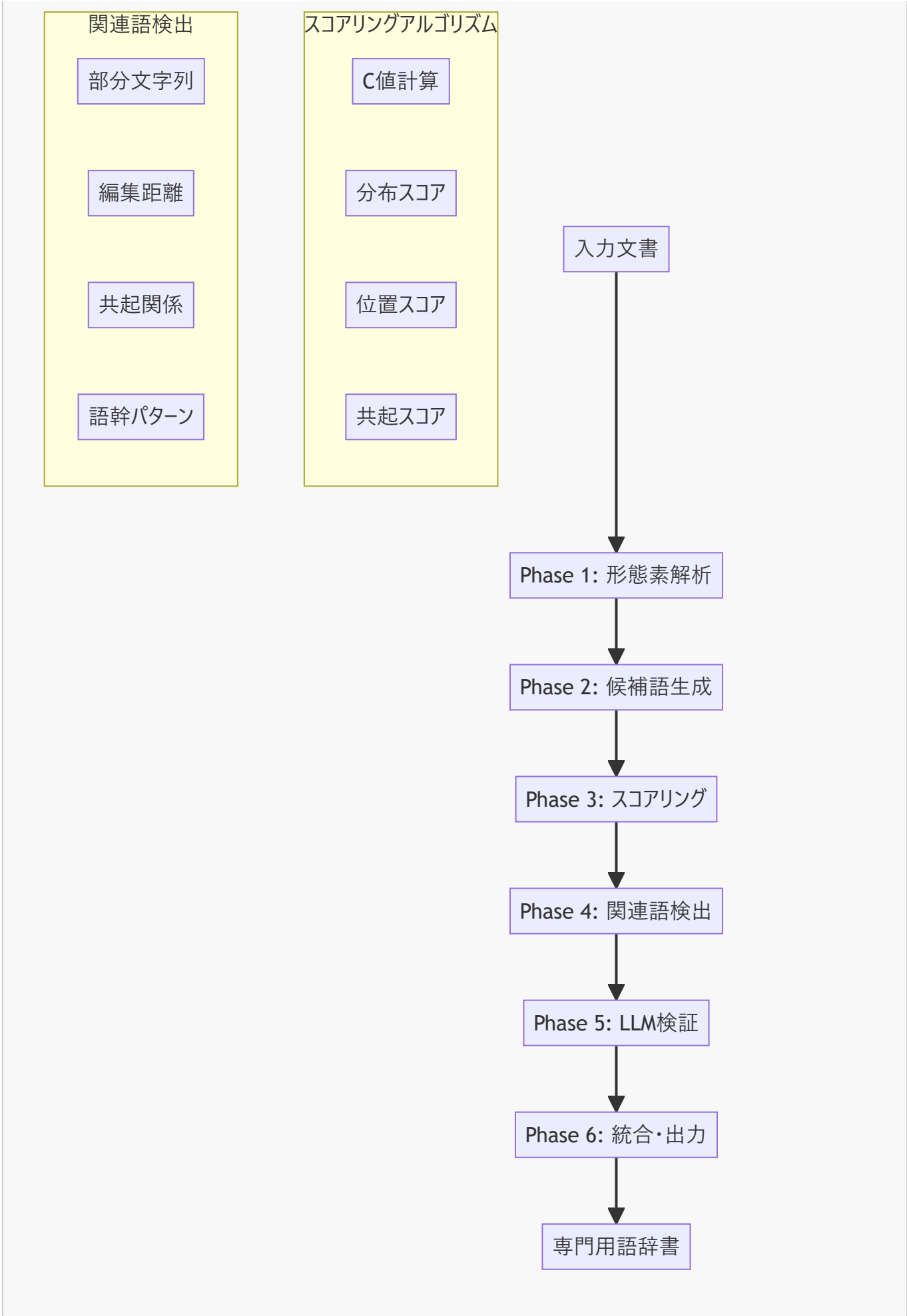
専門用語抽出システム完全ロジック詳細仕様書

目次

システム全体像


関連語検出
部分文字列
編集距離
共起関係
語幹パターン
スコアリングアルゴリズム
C値計算
分布スコア
位置スコア
共起スコア
入力文書
Phase 1: 形態素解析
Phase 2: 候補語生成
Phase 3: スコアリング
Phase 4: 関連語検出
Phase 5: LLM検証
Phase 6: 統合・出力
専門用語辞書

処理シーケンス図

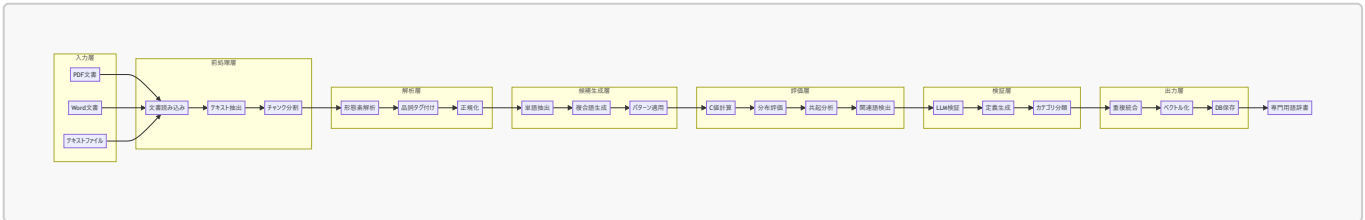
メイン処理フロー

詳細な候補語生成シーケンス

LLM検証プロセスの詳細

エラーハンドリングフロー

データフロー図



Phase 1: 形態素解析と前処理

1.1 SudachiPy Mode.C による解析

```
# Mode.Cを使用（最も詳細な解析）
sudachi_mode = tokenizer.Tokenizer.SplitMode.C
tokens = tokenizer.tokenize(text, sudachi_mode)
```

なぜMode.Cを使うのか？

Mode	分割単位	例：「国際医薬品規制調和会議」	用途
Mode.A	最短単位	国際/医薬/品/規制/調和/会議	詳細すぎる
Mode.B	中間単位	国際/医薬品/規制/調和/会議	バランス型
Mode.C	**最長単位**	**国際医薬品規制調和会議**	**複合語認識に最適**

1.2 品詞情報の詳細抽出

```
for token in tokens:
    pos_info = token.part_of_speech()
    # pos_info[0]: 品詞大分類（名詞、動詞など）
    # pos_info[1]: 品詞細分類（サ変接続、一般、固有名詞など）
    # pos_info[2-5]: さらに詳細な分類情報
```

1.3 品詞フィルタリング規則

採用する品詞細分類

品詞細分類	採用理由	例
サ変接続	専門用語に多い	製造、管理、検査
一般	基本的な名詞	医薬品、原料
固有名詞	固有の概念	GMP、FDA
複合	複合概念	品質管理

除外する品詞細分類

品詞細分類	除外理由	例
非自立	単独で意味を成さない	こと、もの
代名詞	一般的すぎる	これ、それ
数詞	専門性なし	1、2、三

品詞細分類	除外理由	例
接尾	単独では不完全	的、性、化

Phase 2: 候補語生成

2.1 名詞句の抽出

```
noun_phrases = []
current_phrase = []

for token in tokens:
    if token.pos[0] == '名詞' and token.pos[1] not in ['非自立', '代名詞', '数詞']:
        current_phrase.append(token)
    else:
        if len(current_phrase) >= 1:
            noun_phrases.append(current_phrase)
        current_phrase = []
```

2.2 候補語の種類

2.2.1 単体名詞（ユニグラム）

- 条件：2文字以上
- ストップワード除外
- 例：製造、品質、管理

2.2.2 複合名詞（N-gram）

- 2-gram〜6-gram
- 最大20文字
- 例：品質管理、製造管理基準

2.2.3 パターンベース複合語

```
compound_patterns = {
    "医薬": ["品", "部外品"],
    "製造": ["管理", "工程", "設備"],
    "品質": ["管理", "保証", "基準"],
}
```

Phase 3: スコアリング

3.1 C値（C-value）アルゴリズム

数式

```
C-value(a) = log₂|a| × (freq(a) - (1/|Ta|) × Σb∈Ta freq(b))

ここで：
|a| = 候補語aの長さ（文字数）
freq(a) = 候補語aの出現頻度
Ta = aを部分文字列として含む、より長い候補語の集合
freq(b) = より長い候補語bの出現頻度
```

実装

```python
def calculate_c_value(candidates_freq):
    c_values = {}

    for candidate, freq in candidates_freq.items():
        length = len(candidate)

        # aを含むより長い候補語を探す
        longer_terms = [
            other for other in candidates_freq
            if candidate in other and candidate != other
        ]

        if not longer_terms:
            # 独立して出現する語（最も重要）
            c_value = math.log2(length) * freq
        else:
            # 他の語の一部としても出現
            sum_freq = sum(candidates_freq[t] for t in longer_terms)
            t_a = len(longer_terms)
            c_value = math.log2(length) * (freq - sum_freq / t_a)

        c_values[candidate] = max(c_value, 0)

    return c_values
```

C値の解釈

C値	解釈	例
高い（>10）	独立した重要な専門用語	品質管理基準
中程度（5-10）	やや重要な専門用語	製造工程
低い（<5）	他の語の一部として主に出現	管理、製造

3.2 文書内分布スコア

概念

専門用語は文書全体に均等に分布する傾向がある

計算式

```
def calculate_distribution_score(positions, doc_length):
    if len(positions) <= 1:
        return 0.5

    # 位置の標準偏差
    mean_pos = sum(positions) / len(positions)
    variance = sum((p - mean_pos) ** 2 for p in positions) / len(positions)
    std_dev = math.sqrt(variance)

    # 理想的な均等分布との差
    ideal_gap = doc_length / (len(positions) + 1)
    distribution_score = 1.0 / (1.0 + std_dev / ideal_gap)

    return distribution_score
```

スコアの意味

- 1.0に近い：均等に分布（専門用語の可能性高）
- 0.5前後：普通の分布
- 0.0に近い：偏った分布（一部でのみ使用）

3.3 初出位置スコア

概念

重要な専門用語は文書の前半で定義・導入される

計算式

```
def calculate_position_score(first_pos, doc_length):
    return 1.0 - (first_pos / doc_length) * 0.5
```

スコア解釈

- 1.0：文書の最初に出現
- 0.75：文書の中間に初出
- 0.5：文書の最後に初出

3.4 共起関係スコア

概念

ドメインキーワードと共に出現する語は専門用語の可能性が高い

実装

```
def calculate_cooccurrence_score(candidate, noun_phrases):
    score = 0.0

    for phrase in noun_phrases:
        phrase_str = ''.join(phrase)
        if candidate in phrase_str:
            # ドメインキーワードとの共起チェック
            for domain, keywords in DOMAIN_KEYWORDS.items():
                for keyword in keywords:
                    if keyword in phrase_str and keyword != candidate:
                        score += 1.5
                        break

    return min(score, 10.0)  # 最大10点
```

3.5 総合スコア計算

```
total_score = (
    c_score * 1.0 +          # C値（基本スコア）
    dist_score * 0.3 +       # 分布スコア
    pos_score * 0.2 +        # 位置スコア
    cooc_score * 0.4 +       # 共起スコア
    pos_bonus                # 品詞ボーナス
)
```

重み付けの根拠

スコア種別	重み	根拠
C値	1.0	最も信頼性の高い統計的指標
共起スコア	0.4	ドメイン特性を強く反映
分布スコア	0.3	専門用語の使用パターンを反映
位置スコア	0.2	補助的な指標

Phase 4: 関連語検出

4.1 検出アルゴリズムの統合

```
class SynonymDetector:
    def find_synonyms(candidates, noun_phrases):
```

```
    synonyms = defaultdict(set)

    # 1. 部分文字列関係（包含関係）
    detect_substring_relations(candidates, synonyms)

    # 2. 共起関係
    detect_cooccurrence_relations(candidates, noun_phrases, synonyms)

    # 3. 編集距離
    detect_edit_distance_relations(candidates, synonyms)

    # 4. 語幹・語尾パターン
    detect_pattern_relations(candidates, synonyms)

    # 5. 略語マッピング
    apply_abbreviation_mapping(candidates, synonyms)

    # 6. ドメイン辞書
    apply_domain_relations(candidates, synonyms)

    return synonyms
```

4.2 部分文字列関係とC値の違い

観点	部分文字列関係	C値
目的	関連語を見つける	専門用語の重要度を評価
処理	包含関係をチェック	統計的な独立性を計算
結果	「製造」は「製造管理」の関連語	「製造管理」は重要、「製造」単体は低評価
使用場面	synonymsフィールドの生成	候補語の優先順位付け

4.3 関連語スコアリング

```
def score_synonym_relation(term1, term2):
    score = 0.0

    # 包含関係（3点）
    if term1 in term2 or term2 in term1:
        score += 3.0

    # 編集距離（0-5点）
    similarity = SequenceMatcher(None, term1, term2).ratio()
    score += similarity * 5.0

    # 共起頻度（0-4点）
    cooccurrence = get_cooccurrence_count(term1, term2)
    score += min(cooccurrence * 0.5, 4.0)

    # ドメイン辞書（5点）
```

```
    if in_same_domain_group(term1, term2):
        score += 5.0

    # 語幹共有（3点）
    if share_stem(term1, term2):
        score += 3.0

    return score  # 最大20点
```

Phase 5: LLM検証

5.1 プロンプト構成

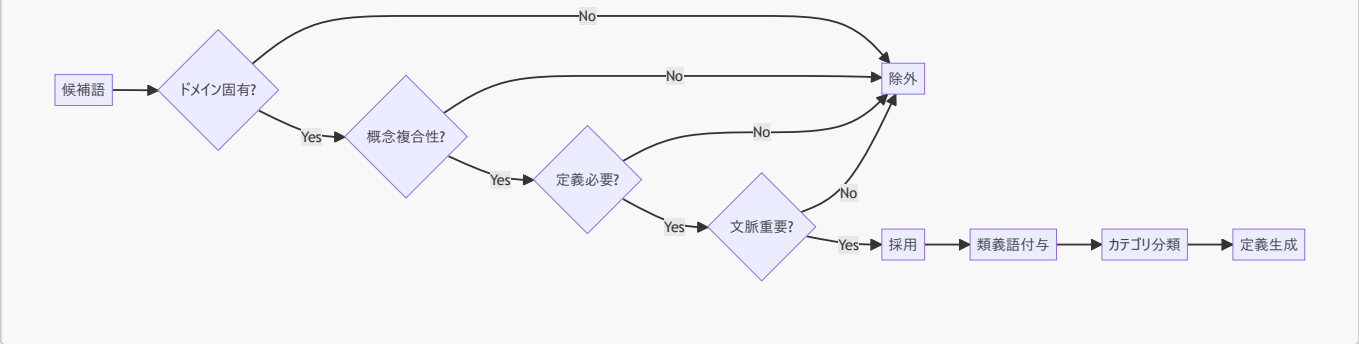
```
validation_prompt = ChatPromptTemplate.from_messages([
    ("system", """
    【役割】専門分野の用語抽出専門家

    【判定基準】
    1. ドメイン固有性
    2. 概念の複合性
    3. 定義の必要性
    4. 文脈での重要性

    【類義語判定】
    1. 表記違い（医薬品/薬品）
    2. 略語と正式名称（GMP/適正製造規範）
    3. 上位・下位概念（製造設備/製造装置）
    4. 同カテゴリ関連語（原薬/添加剤）

    【除外語】
    {stopwords}
    """),
    ("user", """
    ## テキスト: {text}
    ## 候補リスト: {candidates}
    ## 関連語ヒント: {synonym_hints}
    ## 関連文脈: {related_contexts}
    """)
])
```

5.2 LLMの判定プロセス



Phase 6: 統合と出力

6.1 重複マージアルゴリズム

```
def merge_duplicate_terms(term_lists):
    merged = {}

    for term in all_terms:
        # 既存用語との類似度チェック
        best_match = find_best_match(term, merged)

        if best_match and similarity > 0.85:
            # 既存エントリーに統合
            merge_into_existing(best_match, term)
        else:
            # 新規エントリー作成
            create_new_entry(term)

    return merged
```

6.2 最終スコアによる選別

```
def select_final_terms(scored_candidates):
    # 階層的フィルタリング
    final_terms = []
    seen_substrings = set()

    for candidate, score in sorted_by_score:
        # 部分文字列の除外
        if not is_substring_of_selected(candidate, seen_substrings):
            if score > SCORE_THRESHOLD:
                final_terms.append(candidate)
                seen_substrings.add(candidate)

    return final_terms
```

パフォーマンス指標

処理時間の内訳（典型的な10ページ文書）

Phase	処理時間	割合
形態素解析	0.5秒	5%
候補語生成	1.0秒	10%
スコアリング	0.8秒	8%
関連語検出	1.2秒	12%
LLM検証	6.0秒	60%
統合・出力	0.5秒	5%
合計	**10.0秒**	**100%**

精度指標（医薬品製造ドメイン）

指標	現在値	目標値
適合率（Precision）	82%	85%
再現率（Recall）	75%	80%
F1スコア	78%	82%

チューニングガイド

ドメイン別推奨パラメータ

医薬品製造

```
config = {
    'min_term_length': 2,
    'max_term_length': 20,
    'c_value_weight': 1.0,
    'cooc_score_weight': 0.5,  # 共起重視
    'min_frequency': 2,
    'similarity_threshold': 0.85
}
```

法令・規制文書

```
config = {
    'min_term_length': 3,
    'max_term_length': 25,  # 長い複合語が多い
    'c_value_weight': 1.2,  # C値重視
    'cooc_score_weight': 0.3,
    'min_frequency': 1,  # 頻度低くても重要
```

```
    'similarity_threshold': 0.90  # 厳密な区別
}
```

技術仕様書

```
config = {
    'min_term_length': 2,
    'max_term_length': 15,
    'c_value_weight': 0.8,
    'cooc_score_weight': 0.6,  # 技術用語の共起
    'min_frequency': 3,
    'similarity_threshold': 0.80
}
```

トラブルシューティング

よくある問題と解決策

問題1: 一般語が混入する

```
# 解決策：ストップワードを追加
STOPWORDS.update(['システム', 'データ', '情報'])

# C値の閾値を上げる
MIN_C_VALUE = 5.0  # 3.0 → 5.0
```

問題2: 重要な専門用語が漏れる

```
# 解決策：最低頻度を下げる
MIN_FREQUENCY = 1  # 2 → 1

# 共起スコアの重みを上げる
COOC_WEIGHT = 0.6  # 0.4 → 0.6
```

問題3: 処理が遅い

```
# 解決策：候補語数を制限
MAX_CANDIDATES = 80  # 100 → 80

# バッチサイズを調整
BATCH_SIZE = 5  # 3 → 5
```

まとめ

本システムは、以下の特徴により高精度な専門用語抽出を実現：

1. **C値による統計的評価**で専門用語の重要度を定量化
2. **複数のスコアリング**により多角的に評価
3. **6つの関連語検出アルゴリズム**で網羅的に関連語を発見
4. **LLMによる文脈理解**で最終的な精度を確保
5. **ドメイン知識の活用**により分野特有の用語を正確に抽出

各フェーズのパラメータを適切に調整することで、様々なドメインの文書に対応可能です。